REMARKS

Claims 3 and 32 are currently amended. Claims 1-4, 6-13, 15-21 and 23-37 are pending.

Claim Rejection – 35 U.S.C. §102

Claims 9-13 and 15 are rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,193,191 *(McKeeman)*. Applicants respectfully traverse this rejection.

For ease of illustration and organization of arguments, remarks relevant to claims 9-13 and 15 are made in the claim 1 arguments section below.

Claim Rejection – 35 U.S.C. §103

The Examiner rejects claims 1-4, 6-8, 16-21, 23-29, 31-32 and 34-37 under 35 U.S.C. §103(a) as being unpatentable over *McKeeman* in view of "Upgrading Microsoft Visual Basic 6.0 to Microsoft Visual Basic .NET" *(Robinson)*. Applicants respectfully traverse this rejection.

Claim 1

For ease of illustration, claim 1 is discussed first. Claim 1 calls for initiating compilation of a file in a processor-based system in advance of a request from a user to compile the file and detecting the user request to compile the file. Claim 1 also calls for indicating a status of the compilation of the file in response to detecting the user request. Initiating compilation of the file includes compiling the file in response to determining that the file has been modified.

The Examiner's rejection of claim 1 is improper because *McKeeman* and *Robinson*, either alone or in combination as cited by the Examiner, fail to teach all of the claimed features. For example, claim 1 calls for <u>initiating compilation</u> of a file in a processor-based system <u>in advance of</u> a request from a user to compile the file. In the Final Office Action, the Examiner again admits that *McKeeman* does not teach this feature, but the Examiner still argues that *Robinson* teaches <u>initiating compilation in advance of</u> a request. *See* Final Office Action, p.7 (citing *Robinson*, p.16), p.2 Response to Arguments Section. In the Final Office Action, the

Examiner argues that **Robinson** teaches a background "compiler," but "does not mention "parser."" *See* Final Office Action, p.2 Response to Arguments Section. The Examiner's attention is respectfully directed to **Robinson**, p.18, which describes the function of the "compiler" as follows:

> "The result is a true Visual Basic experience, enhanced by background compilation as you type. For example, if you misspell the keyword *Function*, as in
>
> **Funtion myFunction**
>
> as soon as you move off the line, the compiler **parses** it and puts a compiler error in the Task List. It also underlines the word "Funtion" with a blue squiggle indicating the location of the compile error. As soon as you correct the line, the compiler removes the Task List item and erases the underline." (*emphasis added*)

As can be seen, before a user-initiated compile, the "compiler" parses text after it is entered into a file. The passage cited by the Examiner, however, does not teach or suggest <u>initiating compilation</u> of a file in a processor-based system <u>in advance of</u> a request from a user. **Robinson** teaches that a background compiler parses text as it is entered into a file in order to alert the user of syntax errors and the like.

The Examiner's reliance on **Robinson** is misplaced, however, because the parsing is **not** the same as initiating compiling, as would be known to those skilled in the art. Compiling, as would be known to those skilled in the art, would at least involve forming object files from the source code of a project. However assuming *arguendo* that **Robinson** teaches a compiler, the compiler simply performs parsing before a user-initiated compilation. That is, the "compiler" in **Robinson** does not *compile* code until after the user initiates a compilation. **Robinson** does not disclose, *and the Examiner has not cited, any teaching or suggestion in the cited reference* that parsing done by the compiler, as shown in **Robinson**, <u>initiates **compilation in advance** of a request from a user to compile</u>, as called for in claim 1. As such, **Robinson** does not, and cannot, teach this claim feature, and, as admitted by the Examiner, **McKeeman** fails to remedy the fundamental deficiencies of **Robinson**.

Claim 1 also calls for <u>indicating a status of the compilation of the file in response to detecting the user request</u>. In the Final Office Action, Examiner argues that *McKeeman* teaches this claimed feature. *See* Final Office Action, pp. 3 (Response to Arguments) and 7 (Claim Rejections). Applicants respectfully assert that the Examiner, as in the previous Office Actions, improperly characterizes the passages from column 5 of *McKeeman*. In the Final Office Action, the Examiner responds to Applicants' arguments by stating that Applicants refer to features not found in the claims, to wit: "prior to performing the compilation." Applicants respectfully assert that the language of claim 1 recites "initiating compilation of a file…in advance of a request from a user to compile." As such, the claim 1 element of <u>indicating a status of **the compilation** of the file in response to detecting the user request</u> is performed before the user requested compilation (*i.e.*, the user-initiated compilation).

Using this understanding of the claim language as a background, the distinction between the claim 1 and the cited reference can clearly be seen. The Examiner states the cited passage of *McKeeman* teaches indicating a status of the compilation of the file <u>in response to detecting the user request</u>, because "errors are detected and reported." *See* Final Office Action, p.7 This position is untenable. *McKeeman* teaches that **after a user-initiated compilation** of the file, errors in the file may be reported to the user. *See McKeeman*, col. 5, ll. 15-34 (stating "when the developer has reached a point where he wishes to test the code…the compiler 11 is invoked" and *subsequently* "errors are detected and reported."). In other words, *McKeeman* teaches that errors in the file may be reported in response to attempting to compile the file and encountering errors, where the compilation was done **after** a user command to compile has caused a compilation to begin. In contrast, claim 1 recites <u>indicating a status of **the compilation** of the file **in response to detecting the user request**</u>, where "initiating compilation of a file [is done] in advance of a request from a user to compile." The Examiner's attention is respectfully directed to the

Specification for an illustrative, non-limiting example:

> "[I]n accordance with the present invention, because the task processing module 15 may have pre-processed one or more of the tasks associated with the build process, the task processing module 15, upon detecting (at 330) the user request to initiate the build, indicates (at 340) a status of the processing of the one or more tasks." *See* Specification, p.15, line 18 to p.16, line 4.

As this passage clearly illustrates, in the context of claim 1, the pre-user command compilation has already been initialized for at least a portion of a file, so the indication of the status of the compilation may be given in response to detecting the user request. *McKeeman* is not able to provide a status in response to detecting the user request, at least for the reason that there has not been any compiling done when the user input is received. That is, *McKeeman* must detect a user request to compile, *then begin compiling* and then report errors in response to detecting errors during the compilation, which is in contrast to the features of claim 1. It should be noted that any examples from the Specification are for illustrative purposes only, and do not limit the claims in any way.

When properly read in context, the cited passage in *McKeeman* does not teach that the status of the compilation is indicated in response to detecting a user input, as argued by the Examiner. The Examiner's arguments cannot be correct because the compilation in *McKeeman* has not yet taken place when the user request is received. As mentioned above, *McKeeman* teaches that **after a user-initiated compilation** of the file, errors in the file may be reported to the user. *See McKeeman*, col. 5, ll. 15-34. In other words, *McKeeman* fails to teach or suggest indicating the status of the compilation in response to detecting a user input. As such, *McKeeman* does not, and cannot, teach the claimed feature of the status of the compilation is indicated in response to detecting a user input, as called for in claim 1. *Robinson* fails to remedy this fundamental deficiency as *Robinson* is concerned with providing users notifications of possible errors before any compiling is done; that is, *Robinson* is not concerned with any

relationships between compiling and providing status.

Claim 1 also calls for "compiling the file in response to determining that the file has been modified." In the Final Office Action, the Examiner cites *McKeeman*, col. 5, ll. 21-23, as teaching this claimed feature. *See* Final Office Action, p.7. However, as Applicants have stated in the previous Responses, *McKeeman* compiles when the user (developer) decides to compile, **not** in response to a file modification, as called for in claim 1. In the Final Office Action, the Examiner argues that because unchanged files are not compiled, *McKeeman* teaches this claimed feature. The Examiner, however, has not shown how this passage teaches compiling the file in response to determining that the file has been modified, and this is not surprising because *McKeeman* teaches compilation occurs when the user (developer) decides to compile, **not** in response to a file modification, as called for in claim 1. As previously stated, claim 1 recites compiling a file in advance of a request from a user to compile, but in contrast, *McKeeman* teaches that actual compiling is performed after a user-initiated compile command. As such *McKeeman* does not, and cannot, teach the claimed feature of the "compiling the file in response to determining that the file has been modified," as called for in claim 1. *Robinson* fails to remedy this fundamental deficiency as *Robinson* is similarly concerned with compiling upon a user's command/input. As discussed above, *Robinson* parses, but does not compile, prior to a user's compile command.

Without using improper hindsight reasoning and using the claim as a roadmap, the person of ordinary skill in the art would have no apparent reason to modify the references to arrive at the subject matter of claim 1. The Examiner essentially provided a conclusory statement that adding the features of these references together would make for a better product; *i.e.*, the Examiner has simply stated the result of such a combination. *See* Final Office Action, p.4 (stating that "Furthermore, for the sake of argument, even if Robinson only taught parsing, it still provides a

teaching of background computation, which when coupled with the compilation of prior art of record McKeeman's compilation, would teach the claimed limitations."). As such, the Examiner has merely stated that such a combination would have been obvious. However, the Examiner has not pointed to any teachings in the cited references that would **motivate** a person of skill in the art to combine the references. In other words, the question that must be addressed includes "*why* would a person have thought to combine the cited references based on their teachings?", and "*what* was the need?", not simply "what benefits would result?". There must be some motivation or need as to why a combination would have been obvious at the time of the invention.

Applicants respectfully submit that the Examiner's conclusory statement is motivated by improper hindsight and is without support. Applicants respectfully request that the Examiner provide a motivation to combine/substitute that **does not** rely inherently upon the result of such a combination. In other words, a conclusory statement that that "when coupled [the cited references] would teach the claim limitations" is without proper basis and relies entirely upon the result to provide motivation. Applicants respectfully request the Examiner point to a teaching the cited art that shows **where** and **why** a person of skill in the art would have had a need to combine/substitute. In light of the fact that **Robinson** specifically discusses parsing and **McKeeman** is not concerned with background compiling, the Examiner must show some need for background compilation, not merely a result-oriented statement. Motivation to combine aside, as discussed above, even if **McKeeman** and **Robinson** were to be combined, claim 1 as a whole would be untaught and non-obvious over the references.

For at least the aforementioned reasons, claim 1 and its dependent claims are allowable. For at least similar reasons, the remaining independent claims, and their respective dependent claims are also allowable (including claim 9 and its dependent claims).

As such, Applicants request this rejection of claims 1-4, 6-13, 15-21, 23-29 and 31-37 under 35 U.S.C. §103(a) be withdrawn.

Claim 2

Other claims are allowable for additional reasons. For example, claim 2 which depends from method claim 1, recites "wherein initiating compilation of the file comprises compiling the file including one or more code segments to produce an object code file." It should be noted that claim 1 recites initiating compiling in advance of a request from a user to compile, therefore the production of an object code file also occurs in advance of the user request. In the Final Office Action, the Examiner argues that *McKeeman* teaches this claimed feature because *McKeeman* teaches that object code tables are output from a compiler. *See* Final Office Action, p.7 (citing *McKeeman*, col. 5, ll. 30-34). The passage from *McKeeman* relied upon by the Examiner teaches that the output of the compiler (*i.e.*, the object code tables) are produced subsequent to a user-initiated compilation. Applicants respectfully assert that the Examiner has taken the cited passage out of context and improperly applied the passage to the instant claims. Taking the entire passage in proper context, from line 15 to line 34, clearly shows that *McKeeman* describes a compilation initiated by a user (developer). *See McKeeman*, col. 5, ll. 15-17. As such, *McKeeman* does not, and cannot, teach producing an object code file in advance of a request from a user to compile, as called for in claim 1. *Robinson* fails to remedy this fundamental deficiency.

For at least the aforementioned reasons, claim 2 and its dependent claims are also allowable.

Claim 3

Amended claim 3 is discussed next. Claim 3, as amended, depends from method claims 2 and 1, and recites "initiating compilation of a file in a processor-based system in advance of a

request from a user to compile the file further <u>comprises compiling the file to completion</u>." That is, when compilation is initiated, the compilation will compile the file all the way through to its completion. The Examiner admits that *McKeeman* fails to teach the claimed feature of compiling in the background or compiling in advance of a request from a user. *See* Final Office Action, p.7 (with respect to the claim 1 rejection). *Robinson* teaches that a compiler may parse text to flag potential compilation errors as a user edits a file. *See **Robinson***, p.18. *Robinson* does not teach that the file <u>is compiled to completion</u>, as recited in claim 3. *Robinson* teaches that potential compilation errors are flagged on a line-by-line basis during editing. Indeed, *Robinson* does not teach that the file is compiled in advance of a user request at all. The *compiler* taught in *Robinson* parses lines of text and does not <u>compile to completion in advance of a user request</u>, as recited in claim 3. As such, *Robinson* does not, and cannot, teach this claimed feature. *McKeeman* fails to remedy this fundamental deficiency.

For at least the aforementioned reasons, claims 3 and 32, and their respective dependent claims, are also allowable.

Claim 34

Claim 34 is discussed next. Claim 34, which ultimately depends from method claim 24, recites "suppressing at least one of an error and warning that is detected while compiling the modified source files." The Examiner argues that *McKeeman* teaches this claimed feature because *McKeeman* teaches that "errors are detected and reported." *See* Final Office Action, p.15 (citing *McKeeman*, col. 5, ll. 23-34. *McKeeman* teaches that the first error encountered is reported and then compilation stops. This is not "suppressing" errors, as recited in claim 34. In *McKeeman*, there are no additional errors to suppress because *compilation stops* after the first error is encountered. In contrast, claim 34 by virtue of its dependencies recites "wherein initiating the build process comprises performing compiling the modified source files

to produce object code files and linking the object code files to produce executable files" and "suppressing at least one of an error and warning that is detected while compiling the modified source files." As such, *McKeeman* does not, and cannot, teach this claimed feature. *Robinson* fails to remedy this fundamental deficiency.

For at least the aforementioned reasons, claim 34 is allowable.

Claim 30

Claim 30 is rejected under 35 U.S.C. 103(a) as being unpatentable over *McKeeman* and *Robinson*, and further above in view of U.S. Pat. Pub. No 2005/0108682 *(Piehler)*. Applicants respectfully traverse this rejection.

Claim 30 depends indirectly from independent claim 24. Because *McKeeman* and *Robinson* fail to disclose all of the features of claim 24 (for at least the reasons discussed earlier), these references likewise fail to teach the features of dependent claim 30. For at least this reason, claim 30 is allowable.

Claim 33

Claim 33 is rejected under 35 U.S.C. 103(a) as being unpatentable over *McKeeman* and *Robinson* and further in view of *Callahan, II.* Applicants respectfully traverse this rejection.

Claim 33 depends indirectly from independent claim 24. Because *McKeeman* and *Robinson* fail to disclose all of the features of claim 24 (for at least the reasons discussed earlier), these references likewise fail to teach the features of dependent claim 33. For at least this reason, claim 33 is allowable.

Arguments with respect to other dependent claims have been noted. However, in view of the aforementioned arguments, these arguments are moot and, therefore, not specifically addressed. To the extent that characterizations of the prior art references or Applicants' claimed subject matter are not specifically addressed, it is to be understood that Applicants do not

acquiesce to such characterization.

In view of the foregoing, it is respectfully submitted that all pending claims are in condition for immediate allowance. The Examiner is invited to contact the undersigned attorney at (713) 934-4069 with any questions, comments or suggestions relating to the referenced patent application.

Respectfully submitted,

WILLIAMS, MORGAN & AMERSON, P.C.
CUSTOMER NO. 62293

Date: <u>August 10, 2010</u>   By: <u>/Jaison C. John/</u>
                                          Jaison C. John, Reg. No. 50,737
                                          10333 Richmond, Suite 1100
                                          Houston, Texas 77042
                                          (713) 934-4069
                                          (713) 934-7011 (facsimile)
                                     ATTORNEY FOR APPLICANT(S)